

APPENDIX B - SUBSTITUTE SPECIFICATION

The following specification contains a clean-copy version of the amended specification.

**AN APPARATUS AND METHOD FOR PERFORMING SINGLE-
INSTRUCTION MULTIPLE-DATA INSTRUCTIONS**

TECHNICAL FIELD

[0001] The present invention is generally related to performing single-instruction multiple data (SIMD) instructions and, more particularly, is related to an apparatus and method for performing SIMD instructions (e.g., multiply-accumulate operations) using one multiply-accumulate (MAC) unit while minimizing operational latency.

BACKGROUND

[0002] SIMD instructions are those instructions that perform the same operation on two or more pieces of a data word at the same time. A SIMD data word consists of two single-precision floating-point numbers, packed into a floating-point word. In an example of a 82-bit floating-point word, the low-SIMD data is stored in bits 31-0, and the high-SIMD data is stored in bits 63-32. Remaining bits (81-64) of the 82-bit word are set to a predefined constant.

[0003] Currently, two miscellaneous units 5, 6 and two MAC units 3, 4 are used to perform SIMD instructions. Miscellaneous units (MISC) 5, 6 are devices that perform operations not requiring a multiply-accumulate operation, such as, logical functions. A first MAC unit 3 is responsible for performing a multiply-accumulate operation on the high-bits of the SIMD word. The second MAC unit 4 is responsible for performing a multiple-accumulate operation on the low-bits of the SIMD word. MAC unit results are forwarded to a single register file 7. A block diagram of an example of the prior-art system architecture to perform SIMD instructions using multiple MAC units 3, 4 is illustrated in FIG. 1. This prior-art implementation, which includes two full-precision MAC units 3, 4, further includes, two MISC units 5, 6, and two single-precision SIMD units 1, 2. The prior-art system architecture can simultaneously perform any of two SIMD instructions, one SIMD and one non-SIMD instruction, or two non-SIMD instructions.

[0004] Thus, a heretofore-unaddressed need exists in the industry to perform SIMD instructions using a single MAC unit while minimizing operational latency.

SUMMARY

- [0005] The present invention provides an apparatus and method for performing SIMD instructions (e.g., multiply-accumulate operations) using one MAC unit while minimizing operational latency.
- [0006] Briefly described, in architecture, an apparatus for performing single-instruction multiple-data instructions, includes a multiply-accumulate unit configured to generate a data result, the data result having a first half and a second half, a register communicatively coupled to the multiply-accumulate unit, the register configured to store the first half of the data result, and a miscellaneous-logic unit configured to initiate the release of the first half of the data result from the register to synchronize the first half of the data result with the second half of the data result.
- [0007] The present invention can also be viewed as a method for performing SIMD instructions using one MAC unit while minimizing operational latency. The method can be broadly summarized as follows: providing a multiply-accumulate unit configured to generate a first half of a data result and a second half of a data result, applying the first half of the data result at an input of a register, and applying the first half of the data result and the second half of the data result at an input of a buffer when the first half of the data result and the second half of the data result are valid, otherwise applying an exception result at the input of the buffer the first half of the data result and the second half of the data result are invalid.
- [0008] Other features and advantages of the present invention will become apparent to one skilled in the art upon examination of the following drawings and detailed description. It is intended that all such additional features and advantages be included herein within the scope of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

- [0009] The invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

- [0010] FIG. 1 is a block diagram of a prior-art system architecture capable of performing SIMD instructions using multiple MAC units.
- [0011] FIG. 2 is a block diagram of an embodiment of a system architecture capable of performing SIMD instructions using a single MAC unit.
- [0012] FIG. 3 is a flow chart of an embodiment of a method for processing data through the system architecture of FIG. 2.
- [0013] FIG. 4 is an embodiment of a timing diagram illustrating how data is processed through the system architecture of FIG. 2.

DETAILED DESCRIPTION

- [0014] Reference will now be made in detail to the description of the apparatus and method as illustrated in the drawings. While the apparatus and method will be described in connection with these drawings, there is no intent to limit it to the embodiment or embodiments disclosed herein. On the contrary, the intent is to cover all alternatives, modifications, and equivalents within the scope defined by the appended claims.
- [0015] Illustrated in FIG. 2 is an embodiment of a system architecture capable of performing SIMD instructions using a register and a single MAC unit. As shown, register file 21 provides operand A, operand B, and operand C data on operand busses A 22, B 23 and C 24. Operand busses A 22, B 23 and C 24 transfer operand data from the register file 21 to logic 32 in MISC 31 and to the MAC 41. There is no logic in MISC 31 between register file 21 and MAC 41 for the operands. The MAC 41 receives operand A, operand B, and operand C data on operand busses A 22, B 23, and C 24, respectively, in logic 42. Logic 42 also receives operational control codes from an external control unit (not shown for simplicity of illustration). Operational control codes are used to compute a desired result.
- [0016] MISC logic 32 uses the operand A, operand B, and operand C data and operational control codes to generate a series of four sets of result control signals and their complements to control the various bus drivers in both MISC 31 and MAC 41.
- [0017] Result control signal A is generated in accordance with the following expression: $A = \text{miscop} + \text{macop} * \text{misc_result} + \text{simdop}$, where miscop indicates

that there is an instruction for MISC 31, macop indicates that there is an instruction for MAC 41, misc_result indicates that there is a non-SIMD MAC 41 instruction that contains MISC 31 generated result(s), and simdop indicates that there is a SIMD instruction for either MISC 31 or MAC 41. Generating signal A configures data bus 36 to transmit data to result data bus 71A. Data bus 36 transmits data to result data bus 71A when signal 72 enables buffer/driver 33.

[0018] Result control signal B is generated in accordance with the following expression: $B = \text{miscop} + \text{macop} * !\text{simd} * \text{miscresult} + \text{macop} * \text{misc_result_high} * \text{simdhigh}$, where misc_result_high is a SIMD MAC 41 instruction with a MISC 31 result on the high-half data bits (i.e., bits 63-32), and simdhigh is the result of the SIMD operation on the high bits. Generating signal B configures data bus 37 to transmit data to the high-half result data bus 61A. Data bus 37 transmits data to the high-half result data bus 61A when signal 62 enables buffer/driver 34. The high-half result data bus 61A transmits data to register 80 for storage. Register 80 stores the first half of the data result while the second half of the data result is being computed. MISC logic 32 determines when to release the first half of the data result stored in register 80 to synchronize the first half of the data result with the second half of the data result.

[0019] Result control signal C is generated in accordance with the following expression: $C = \text{miscop} + \text{macop} * !\text{simd} * \text{miscresult} + \text{macop} * \text{misc_result_low} * \text{simdhigh}$, where misc_result_low is the MISC31 result on the low data bits (i.e., bits 31-0). Generating signal C configures data bus 38 to transmit data to the result data bus 51A. Data bus 38 transmits data to the result data bus 51A when signal 52 enables buffer/driver 35.

[0020] Result control signal D is generated in accordance with the following expression: $D = \text{macop} * !\text{misc_result_low} * \text{simdhigh}$. Generating signal D configures data bus 61B to transmit data from register 80 to result data bus 51B. Data bus 61B transmits data to result data bus 51B when signal 75 enables buffer/driver 27.

[0021] Result control signals A-D are valid in MISC 31 and MAC 41 during period x and period y on the timing diagram (FIG. 4). Signals A-D are floating point control unit signals that are qualified by the simdhigh/simdlow signals. In SIMD

mode; result control A is valid during period y, result control B is valid for both periods x and y, and result control signals D&C are valid for period y. In non-SIMD mode, result control signals A-D are valid during floating-point clock stage FP4.

[0022] The result control signals listed above are generated in accordance with the following instructions and signals:

miscop = an instruction exists for MISC 31;
 macop = an instruction exists for MAC 41;
 misc_result = a non-SIMD MAC instruction that contains results generated by MISC 31;
 misc_result_low = a SIMD MAC instruction that contains low-half data generated by MISC 31;
 misc_result_high = a SIMD MAC instruction that contains high-half data generated by MISC 31;
 simdhigh = asserted during clock stage FP4 (high-operand) during which, the results for the high-half SIMD are generated. (It is assumed that the signal simdhigh is only active when signal simd is active);
 simd = a SIMD instruction exists (for either MISC 31 or MAC 41).

[0023] These signals are generated by the MISC 31, based upon the operational control codes and operands. The operands are received by MISC 31 from register file 21. The operational control codes come from an external control unit (FPU Control) (not shown) that communicates with the main instruction fetch unit. The FPU Control and MISC 31 units are responsible for the correct staging of pipelined control information.

[0024] Bus drivers 27, 33, 34, and 35 in FIG. 2 will drive only when their enable line is asserted. An example will be illustrated with regard to Fig. 3 herein described in detail below. The present apparatus is not limited to the bus control methodology illustrated and described in association with FIG. 2. Other methods of performing the bus multiplexing are possible, such as repeating multiplexers, etc.

[0025] Note that the apparatus illustrated in FIG. 2 contains one functional unit. Those skilled in the art should understand that a second functional unit can be arranged to interface with register file 21, as shown in the system architecture of

FIG. 1. However, the apparatus illustrated in FIG. 2 is able to simultaneously perform two SIMD instructions (4 total operations), one SIMD and one non-SIMD instruction, or two non-SIMD instructions. While the embodiment illustrated in FIG. 2 requires an additional cycle of latency than that of the prior-art system illustrated in FIG. 1, the apparatus of FIG. 2 enables a SIMD instruction to be executed in parallel with another instruction (SIMD or non-SIMD).

[0026] FIG. 3 is a flow chart of an embodiment of a method for processing data through the system architecture of FIG. 2. First, the register file 21 (FIG. 2) drives data on operand busses A 22, B 23, and C 24, for two consecutive cycles. During the first cycle, as illustrated in block 101, MAC 41 latches the low-operand data into low-data latches in logic 42, which is now prepared to begin operations on the next cycle. Operational control code information arrives from the external control unit prior to or concurrently with the first clock cycle.

[0027] During the second cycle, as illustrated in block 102, MAC 41 starts operations on the low-operand data and latches the high-operand data into the high-data latches of logic 42. MISC 31 latches both high and low-operand data and operational control codes arrive via busses A 22, B 23, and C 24.

[0028] During the third cycle, as illustrated in block 103, MAC 41 continues operation on the low-operand data and starts operation on the high-operand data. The MISC 31 begins its operation on both the high and low-operand data. A second instruction (either SIMD or non-SIMD) may have its operands and/or operational control codes delivered to the MISC 31, while MAC 41 starts on the next cycle.

[0029] During the fourth cycle, as illustrated in block 104, MAC 41 continues operation on both the lower and higher-operand data. A third instruction can also enter the busses A 22, B 23, and C 24 during this cycle. This is a fully pipelined system and once the instructions leave a certain clock stage (e.g., FP1, FP2, FP3, FP4, WRB) another SIMD or non-SIMD instruction can enter that clock stage.

[0030] During the fifth cycle, as illustrated in block 105, MAC 41 delivers the low-operand data result onto the high-half result data bus 47. The low-operand data result is then transmitted to the high-half result data bus 61A. This is accomplished by applying signal 62 from logic 32 as an input at inverter 63 to generate enable signal 64. Enable signal 64 commands buffer/driver 44 to transmit lower-operand

data from the high-half result data bus 47 to the high-half result data bus 61A.

Signal 62 is also input in its original value into buffer/driver 34. This original value for signal 62 disables buffer/driver 34 from transmitting operand-data result from logic 32 onto high-half result data bus 61A. The low-operand data result from the high-half result data bus 61A is latched into register 80. Concurrently, during the fifth cycle, MAC 41 continues to operate on the high-operand data.

[0031] During the sixth cycle, as illustrated in block 106, MISC 31 indicates whether to use the MAC 41 results or the MISC 31 exceptional results. MISC 31 indicates which results are to be utilized by generating signals on signal lines 52, 62, 72 and 75, respectively. These signals cause the appropriate bus drivers 25-27, 33-35, 43-45, 53, 63 and 73 to place result data on result bus 51A, 61A, or 71A as desired. MISC 31 generates the following signals, illustrated in the table below, on signal lines 52, 62, 72, and 75, respectively, to command the appropriate bus drivers to place result data on result bus 51A, 61A, or 71A.

[0032] Cases 1-4 in Table I below are SIMD MAC operation cases. The cases are as follows:

CASE	EXCEPTIONS	BUFFER/DRIVERS ON	SIGNALS ACTIVE
Case 1	No exceptions	27, 33, 44	64, 72, 75
Case 2	Low exception	26, 33, 35, 44	52, 64, 72, not 75
Case 3	High exception	27, 33, 34	62, 72, 75
Case 4	Both exceptions	26, 33, 34, 35	52, 62, 72, not 75
Case 5	Non-SIMD, no exception	26, 43, 44, 45	54, 64, 74, not 75
Case 6	Non-SIMD, exception	26, 33, 34, 35	52, 72, not 75
Case 7	MISCOP	26, 33, 34	62, 72, not 75

Table I

[0033] If the MISC 31 does not detect an exceptional case for the high mantissa, the MAC 41 delivers the high-operand data result onto the high-half result data bus 61A. If the MISC 31 does not detect an exceptional case for the low mantissa,

register 80 drives the lower-operand data result onto the lower-half result data bus 51A.

[0034] Whenever MISC 31 detects an exception, MISC 31 delivers the result. In any of the SIMD cases, MISC 31 delivers the exponent result. MISC 31 delivers the exponent result from buffer/driver 33 by generating a signal on signal line 72.

[0035] During the seventh cycle, as illustrated in block 107, the combined result is written to the register file 21 (FIG. 2). While the system architecture illustrated and described utilizes a four clock period latency other clock cycle latencies are possible.

[0036] FIG. 4 is an embodiment of a timing diagram illustrating how data is processed through the system architecture of FIG. 2. As illustrated in FIG. 4 a plurality of non-SIMD operations 112 and SIMD operations 113 are controllably configured via result control signals A-D in accordance with low-operand data states 114 and high-operand data states 115 that correspond to clock signal trace 111. Register file 21 (FIG. 2) drives low-operand data and high-operand data on the operand busses A 22, B 23, and C24 (FIG. 2), over three consecutive clock cycles as shown by signal traces 131 and 132. The arrival of low-operand data, during clock stage FP1 for low-operand data is shown by signal trace 131. The arrival of high-operand data, during clock stage FP1 for high-operand data is indicated by signal trace 132.

[0037] As indicated by signal trace 133, a low-operand result is calculated during clock stages FP2 and FP3 and latched during clock stage FP4 for low-operand data. Signal trace 134 illustrates that a high-operand result is calculated during clock stages FP2 and FP3 and latched during clock stage FP4 for high-operand data. The apparatus of FIG. 2 is a fully pipelined system and once the instructions leave a clock state (e.g., FP1, FP2, FP3, FP4, WRB) another instruction, SIMD or non-SIMD can enter that clock stage. MISC 31 generates signals 52, 62 and 72, which control the output buffer/drivers 43, 44 and 45 (FIG. 2). Output buffer/drivers 43, 44, and 45, control the data transmitted along result data busses 51A, 61A, and 71A (FIG. 2), respectively during clock cycle 5.

[0038] During clock cycle 5, MAC 41 delivers the low-operand data result 133 onto the high-half result data bus 47(FIG. 2). Signals on data busses 46, 47, and 48 (FIG.

2), generated by MAC 41 are dependent upon the operand and the operational control codes applied on operand busses A 22, B 23, and C 24. For SIMD instructions, high-half result data bus 47 is the most significant output bus. For non-SIMD instructions, all three busses (*i.e.*, 46, 47 and 48 (FIG. 2)) are significant output busses. Low-operand data result is transmitted to the high-half result data bus 61A. Thereafter, low-operand data result from the high-half result data bus 61A is latched into register 80 (FIG. 2). Also, during clock cycle 5, MAC 41 (FIG. 2) continues to operate on the high-operand data as indicated by signal trace 134.

[0039] During clock cycle 6, MAC 41 (FIG. 2) delivers the high-operand data result as indicated in signal trace 134 onto the high-half result data bus 47. MISC 31 indicates whether MAC 41 results or the MISC 31 exception results are applied to high-half result data bus 47. In addition, SIMD results are now available as illustrated by signal trace 136.

[0040] During clock cycle 7, the combined result is written to the register file 21 (FIG. 2) as indicated by signal trace 135.

[0041] It should be emphasized that the above-described embodiments of the present invention, particularly, any "preferred" embodiments, are merely possible examples of implementations, merely set forth for a clear understanding of the principles of the invention. Many variations and modifications may be made to the above-described embodiment(s) of the invention without departing substantially from the principles of the invention. All such modifications and variations are intended to be included herein within the scope of the present invention and protected by the following claims.